Synthesis of Multiple-Valued Logic Networks Based on Tree-Type Universal Logic Module

doi: 10.1109/TC.1977.1674797

V. Conclusion

The maximum number of thresholds realized by a multi-threshold tree network has been obtained and it has been seen that each of these thresholds can be set arbitrarily. In other words, the tree network with k elements can realize any $(2k-1)$-threshold logic function. Therefore it can be said that the various tree networks, which contain the cascade and the two-level networks, with the same number of elements are equivalent to each other in view of realized logic functions. Thus we can synthesize the network in which the number of internal inputs of each element and the number of levels are chosen appropriately. The relation between the realized thresholds and the parameters of each element, i.e., weights and a threshold, may be derived by the proof of Property 4 and the example.

Acknowledgment

The author wishes to thank Prof. Koosuke Harada, the Department of Electronics, Kyushu University, Fukuoka, Japan, for his helpful comments and discussions.

References

[1] D. R. Haring, "Multi-threshold threshold elements," *IEEE Trans. Electron. Comput.*, vol. EC-15, pp. 45–65, Feb. 1966.
[2] S. Ghosh and A. K. Choudhury, "Cascaded multithreshold networks," *IEEE Trans. Comput.*, vol. C-20, pp. 655–662, June 1971.
[3] A. Imamiya, S. Noguchi, and J. Oizumi, "On properties of threshold networks," *Trans. Inst. Electron. Commun. Eng. (Japan)*, vol. 55-D, pp. 699–706, Nov. 1972.
[4] O. Ishizuka, "Maximum numbers of thresholds realized by multi-threshold networks," *Trans. Inst. Electron. Commun. Eng. (Japan)*, vol. 58-D, pp. 365–372, July 1975.
[5] C. L. Sheng, "A graphical interpretation of realization of symmetric Boolean functions with threshold logic elements," *IEEE Trans. Electron. Comput.*, vol. EC-14, pp. 8–18, Feb. 1965.

Synthesis of Multiple-Valued Logic Networks Based on Tree-Type Universal Logic Module

MICHITAKA KAMEYAMA AND TATSUO HIGUCHI

***Abstract*—This correspondence presents a theoretical study on the synthesis of multiple-valued logic networks based on tree-type universal logic modules (T-ULM's). The mathematical notation of T-ULM is introduced. On the basis of the mathematical properties, an algorithm for synthesizing an arbitrary logic function of n variables with a smaller number of modules is presented. In this algorithm, only true and constant inputs are allowed at input terminals. The algorithm consists of two parts. The first one is a functional decomposition, and the other one is the proper order of the expansion which is the problem of finding the most incomplete tree structure. Thus it is established that the systematic and nonexhaustive procedure of the algorithm gives a suboptimal solution for the reduction of the number of T-ULM's.**

***Index Terms*—Compatible set, complex disjunctive decomposition, control variables, dependency, expansion, residue functions, row multiplicity, ternary T-gate, tree-type universal logic module (T-ULM), true and constant inputs.**

Manuscript received December 1, 1975; revised June 30, 1976 and January 18, 1977. This paper was partly presented at the 1975 International Symposium on Multiple-Valued Logic, Bloomington, IN, May 1975.

The authors are with the Department of Electronic Engineering, Faculty of Engineering, Tohoku University, Sendai, Japan.

I. Introduction

In the binary logic, the recent development of integrated circuit technology has made possible using the complex modules as building blocks in place of the NAND or NOR gates. These modules can realize any logic function of up to a fixed number of variables. Such modules are called universal logic modules (ULM's) [1]–[4].

In the multiple-valued logic, there are various type of operators such as in Post algebra which constructs a complete system [5]. However, the logic design is not so easy as in the binary logic. Moreover, the number of operators necessary to realize any logic function of n variables increases, particularly in the operators which construct a complete system only with a single gate such as Sheffer stroke functions [6], [7]. Under these criteria, the use of ULM's in the multiple-valued logic seems to be attractive in practical applications. Some authors [8]–[11] have studied ULM's in the multiple-valued logic. Their interests center about the number of input/output terminals of the module. The ternary T-gate which is called "conditioned disjunction" by logicians is also this kind of modules [12]–[15]. The modules such as T-gates are so-called tree-type universal logic modules (T-ULM's) [3].

However, neither the mathematical properties of T-ULM's nor the minimization procedure of the network of T-ULM's have been studied satisfactorily so far. Yau and Tang [3] present the three possible methods for the reduction of the number of T-ULM's in the binary logic, but its systematic and nonexhaustive procedure has not been obtained.

In this correspondence, the mathematical notation of T-ULM which is the extension of the one of T-gate is introduced. Based on this notation, the mathematical properties of T-ULM's in regard to a canonical expansion, various manipulations of a functional expression are discussed. Then, an algorithm for synthesizing a logic network of any large number of variables is presented [16]. In the algorithm, only true and constant inputs are allowed. The algorithm consists of two parts; the first one is on the functional decomposition which is the problem of how to decompose a logic function using a minimum number of T-ULM's. The other one is the proper order of the expansion which is the problem of finding the most incomplete tree structure. In each part, the evaluation function available to synthesizing a network with a minimum number of T-ULM's is discussed.

II. Mathematical Properties of T-ULM's

In the r-valued logic system, the finite set of r logic values is defined as $L = \{0,1,\cdots,r-1\}$. T-ULM with k control variables is defined as follows.

Definition 1: Let $C = (c_1,c_2,\cdots,c_i,\cdots,c_k)$ be the control vector whose elements are k control variables. The number of the distinct states of the vector C is r^k, and let the scalar S be equivalent to $\Sigma_{i=1}^{k} c_i r^{i-1}$ (denoted by $C \leftrightarrow S$). Let the set P be $(p_0,p_1,\cdots,p_j,\cdots,p_{r^k-1})$, which is called the set of residue functions. The output of T-ULM, U_{out} is defined as

$$U_{\text{out}} = U(p_0,p_1,\cdots,p_j,\cdots,p_{r^k-1};c_1,c_2,\cdots,c_i,\cdots,c_k) = U(P;C).$$

If $C \leftrightarrow S = j$, then $U_{\text{out}} = p_j$.

The symbol of T-ULM is shown in Fig. 1.

We shall present some of the useful mathematical properties of T-ULM's. The proofs of these properties are presented in [16].

Lemma 1: Let $\boldsymbol{a}$ be the element of the set L. Given $\boldsymbol{l}$ such that $\boldsymbol{l} \in (1,\cdots,k)$. If $c_{\boldsymbol{l}} = \boldsymbol{a}$, then $m_{\boldsymbol{a}}$ is defined as $m_{\boldsymbol{a}} = \Sigma_{i=1}^{k}{}' c_i r^{i-1} + \boldsymbol{a} r^{\boldsymbol{l}-1}$ (the number of distinct values of $m_{\boldsymbol{a}}$ is r^{k-1}), where Σ' denotes the summation except $i = \boldsymbol{l}$. Let $g(x)$ be a unary function in the r-valued logic. If $p_{m_{\boldsymbol{a}}} = g(\boldsymbol{a})$ is hold for all the values of $\boldsymbol{a}$, then $U(P;C) = g(c_{\boldsymbol{l}})$.

Lemma 2: If the elements of the set P are all the same, i.e., $p_0 = p_1 = \cdots = p_{r^k-1}$, then $U(P;C) = p_0$ for all the values of c_i.

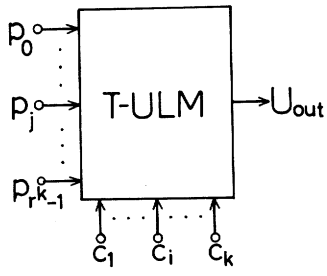

Fig. 1. Symbol of T-ULM.

Lemma 3: Let X be the set of n variables $(x_1,x_2,\cdots,x_n)$, and X_c be the set of k variables in X. Then $U(p_0(X),\cdots,p_j(X),\cdots,p_{r^k-1}(X);X_c) = U(p_0(X',X_c \leftrightarrow 0),\cdots,p_j(X', X_c \leftrightarrow j),\cdots,p_{r^k-1}(X', X_c \leftrightarrow r^k-1); X_c)$, where $X' \cap X_c = \phi$ (empty set) and $X' \cup X_c = X$.

Theorem 1: If constant and true inputs are allowed, an arbitrary function of n variables can completely be expressed with a maximum of $(r^{bk}-1)/(r^k-1)$ T-ULM's, where $b = \lfloor n/k \rfloor$ ($\lfloor x \rfloor$ denotes the smallest integer such that $\lfloor x \rfloor \geq x$).

Theorem 2—Canonical Expansion Theorem: An arbitrary function of n variables can always be expressed in the following form: $f(X) = U(p_0(X'), p_1(X'),\cdots, p_j(X'),\cdots, p_{r^k-1}(X');X_c)$, where $p_0(X') = f(X',X_c \leftrightarrow 0),\cdots, p_j(X') = f(X', X_c \leftrightarrow j),\cdots, p_{r^k-1}(X') = f(X',X_c \leftrightarrow r^k-1)$.

Lemma 4: A function of n variables is expressed in a complete tree structure. Let U_1 be the T-ULM of the first level which is the nearest to the output terminal. Similarly let U_l be the T-ULM of the lth level. Then the number of U_l's is $r^{k(l-1)}$, where $1 \leq l \leq \lfloor n/k \rfloor$.

A function of n variables is expressed in a canonical tree structure as shown in Fig. 2.

Theorem 3: Let $g(x)$ be a unary function in the r-valued logic system. Then $g(U(P;C)) = U(g(P);C)$, where the jth element of $g(P)$ is $g(p_j)$.

Theorem 4: For each control variable $c_i (i = 1,\cdots,k)$ a unary function $g_i(x)$ is given. Then $U(P;g_1(c_1),g_2(c_2),\cdots,g_k(c_k)) = U(p_{0'},\cdots,p_{j'},\cdots,p_{(r^k-1)'};C)$, where $j' = \sum_{i=1}^{k}(g_i(c_i))r^{i-1}$.

III. An Algorithm for Synthesizing a Network of T-ULM's

Since Theorem 1 assures functional completeness in a network of T-ULM's, any logic function of several variables is realized with T-ULM's of few control variables in a tree structure. However, as the number of variables of a function becomes large, the number of T-ULM's necessary to realize the function exponentially increases in a canonical tree structure. The technique for the reduction of the number of T-ULM's is available especially in the function of large variables.

A. Functional Decomposition

There are several types of decomposition. In this section, a complex disjunction decomposition is applied into the synthesis of a T-ULM network. The concept of the maximal compatible sets in the multiple-valued logic [17]–[19] is introduced.

Definition 2: Let $f(X)$ be a function of n variables, and let a set of n variables be divided into two disjunctive subsets Y(p variables) and Z($n-p$ variables), where $Y \cap Z = \phi$, $Y \cup Z = X$. A decomposition table is constructed with the variables in Y defining the rows and the variables in Z defining the columns. Then the total number of different vectors in the rows is defined as "row multiplicity," i.e., $K(Y;f)$. The total number of different vectors in the columns is defined as "column multiplicity," i.e., $K(Z;f)$.

Let define the evaluation function by which the decomposition is available to the minimization of a T-ULM network. The problem is how to find Y and Z effective to the minimization. In order to consider such things, a lemma is shown.

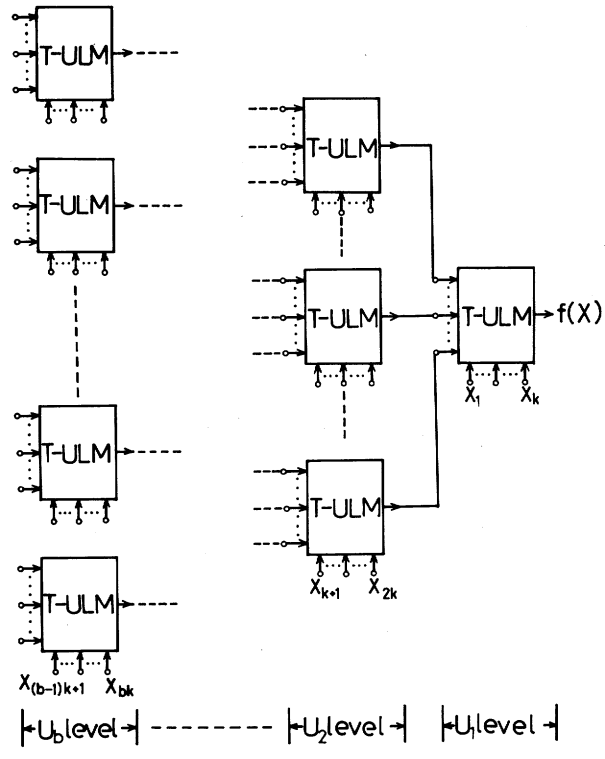

Fig. 2. Canonical tree structure.

Lemma 5: Let $f(X)$ be decomposed as the following form.

$$f(X) = g(\alpha_1(Y),\alpha_2(Y),\cdots,\alpha_u(Y),Z). \tag{1}$$

The minimum value of integer u is given as $u = \lfloor \log_r K(Y;f) \rfloor$.

If $u = 1$ and $p \geq 2$, then (1) is called a simple disjunctive decomposition. Equation (1) corresponds to the block diagram of Fig. 3.

Let M_n be the upper bound on the number of T-ULM's necessary to realize a function of n variables. By Theorem 1, $M_n = (r^{\lfloor n/k \rfloor k}-1)/(r^k-1)$. In (1), g is a function of $u+n-p$ variables and each α_i is a function of p variables. Therefore, the total upper bound on the cost of the function in (1) can be expressed as

$$\begin{aligned} F &= u \cdot M_p + M_{u+n-p} \\ &= u \cdot (r^{\lfloor p/k \rfloor k}-1)/(r^k-1) + (r^{\lfloor (u+n-p)/k \rfloor k}-1)/(r^k-1). \end{aligned} \tag{2}$$

It is reasonable that Y and Z which minimize F in (2) are selected. However, if the minimum value of F is greater than M_n, then the decomposition is not effective to the reduction of the number of T-ULM's, because the decomposition makes the upper bound on the cost of the given function increase more. From this point of view, the decomposition is carried out if $F \leq M_n$ for the minimum F.

For an allowed decomposition, the functions $\alpha_1,\alpha_2,\cdots,\alpha_u$ are specified as follows. If the vectors of the rows are the same for each other, then the same codes of length u are assigned into the corresponding rows, that is, the same code of $\alpha,\cdots,\alpha_u$ is assigned into each compatible set.

Example 1: Consider the ternary function of Table I. Let the function be synthesized with T-ULM's of one control variable. If $Y = (x_1,x_2)$ and $Z = (x_3)$, then $\{R_0,R_5,R_7\}$, $\{R_1,R_3,R_8\}$, and $\{R_2,R_4,R_6\}$ are maximum compatible sets. Therefore, $K(Y;f) = 3$. In this case, the minimum value of F is obtained as $F = (3^2-1)/2 + (3^2-1)/2 = 8$ associated with the partition $Y = (x_1,x_2)$ and $Z = (x_3)$. On the other hand, $M_3 = (3^3-1)/2 = 13$. So, the de-

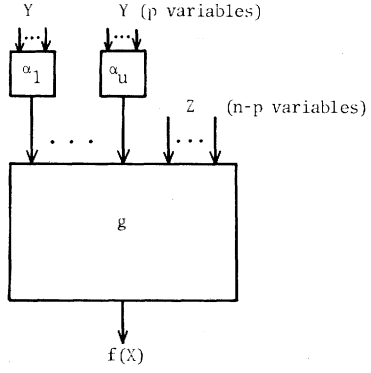

Fig. 3. Block diagram of the decomposition.

TABLE I
TRUTH TABLE OF EXAMPLE 1

	x_1, x_2 \ x_3	0	1	2
R_0	0 0	0	1	2
R_1	0 1	1	2	0
R_2	0 2	2	0	1
R_3	1 0	1	2	0
R_4	1 1	2	0	1
R_5	1 2	0	1	2
R_6	2 0	2	0	1
R_7	2 1	0	1	2
R_8	2 2	1	2	0

composition of $Y = (x_1,x_2)$ and $Z = (x_3)$ is allowed. As a result, the functions $\alpha_1(x_1,x_2)$ and $g(\alpha_1,x_3)$ are given as shown in Table II.

Once such decomposition has been allowed, and $(\alpha_1,\cdots,\alpha_u)$ and g are specified, then the functions $\alpha_1,\cdots,\alpha_u$ and g are each tested to determine if the decomposition is allowed. In this way, the decomposition is carried out iteratively until the decomposition is not allowed. These procedures are summarized in Subalgorithm 1.

Subalgorithm 1:

Step 1: Divide the given n variables into two disjunctive subsets Y and Z.

Step 2: Calculate "row multiplicity" $K(Y;f)$ and $u = \lfloor \log_r K(Y;f) \rfloor$.

Step 3: Calculate $F = u \cdot M_p + M_{u+n-p}$.

Step 4: After all the possible pairs of Y and Z are checked, select Y and Z which minimize the value F.

Step 5: If $F \leq M_n$, go to Step 6. If $F > M_n$, go to Subalgorithm 2.

Step 6: Assign the same code into the compatible set. Specify the functions $\alpha_1,\cdots,\alpha_u$ and g.

Step 7: The procedures from Step 1 and Step 6 is applied into $\alpha_1,\cdots,\alpha_u$ and g, respectively.

B. Proper Order of Expansions

As the number of input variables increases, the number of possible ways of expansion becomes enormously large. So, a technique for the proper order of expansion must be found. In order to get a minimum realization of a T-ULM network, a new concept "dependency" is defined.

Definition 3: Let X_c^i be a subset of a set of variables $X = (x_1,x_2,\cdots,x_n)$. X_c^i has the number a of elements, where $a \leq k$. Let X' be a set such that $X' \cap X_c^i = \phi$ and $X' \cup X_c^i = X$. For the set X', there exist r^{n-a} vertices. Given an arbitrary function of n variables, $f(X) = f(X',X_c^i)$. $N^f(X_c^i)$ is defined as the number of vertices in X' such that $f(X',X_c^i) \neq$ constant and $f(X',X_c^i) \neq x_j$, where $x_j \in X_c^i$. $N^f(X_c^i)$ is called "dependency" of a function $f(X)$ for X_c^i.

TABLE II
TRUTH TABLE OF α_1 AND g

$\alpha_1(x_1, x_2)$

x_1 \ x_2	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$g(\alpha_1, x_3)$

α_1 \ x_3	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Example 2—$r = 2$, $k = 2$: Consider the function $f(x_1,x_2,x_3,x_4)$ of Table III. Let X_c^1 and X_c^2 be (x_1,x_2) and (x_3,x_4), respectively. Since $f(0,0,x_3,x_4) = 1$, $f(0,1,x_3,x_4) = x_3$, $f(1,0,x_3,x_4) = x_4$ and $f(1,1,x_3,x_4) = 0$, $N^f(X_c^2) = 0$. Similarly, $N^f(X_c^1) = 4$.

We shall present some properties of "dependency" whose proofs have been given in [16].

Theorem 5: Let X be divided disjunctively to the number b of subsets $(X_c^1,\cdots,X_c^i,\cdots,X_c^b)$, where $b = \lfloor n/k \rfloor$. Let $f(X)$ be expressed as

$$f(X) = U(f_0(X'),f_1(X'),\cdots,f_j(X'),\cdots,f_{r^k-1}(X');X_c^i).$$

In the total sum of "dependencies", the following relation is held for any i.

$$\sum_{I=1}^{b} N^f(X_c^I) = N^f(X_c^i) + \sum_{m=0}^{r^k-1} \sum_{I=1}^{b}{}' N^f m(X_c^I),$$

where Σ' denotes the summation except i.

Theorem 5 assures that if subsets $(X_c^1,\cdots,X_c^i,\cdots,X_c^b)$ are given, then the sum of "dependencies" of each residue function and "dependency" of the output function $f(X)$ for the subset X_c^i is constant for any i.

Theorem 6: Let a function of n variables be expressed in a tree structure, and let the number of levels be b. In one U_I of the Ith level, "dependency" of the value $r^{(b-I)k}$ can be admitted at its maximum.

Theorem 7: Consider the case of $r \neq 2$ or $k \neq 1$, and let a function $f(X)$ be completely specified. If "dependency" $N^f(X_c^i)$ for an arbitrary X_c^i which is the subset of k elements in X becomes zero, then this function $f(X)$ is realized only with a constant function or a true input.

Lemma 6: Consider the case of $b = \lfloor n/k \rfloor = 2$. Suppose that X_c^i is selected for which "dependency" $N^f(X_c^i)$ becomes minimum. If X_c^i is assigned to the second level, and the other is assigned to the first level, then it is the minimum realization (assume that identical residue functions cannot be used as common inputs).

Example 3: Consider the synthesis of the function of Table III using T-ULM's of two control variables. By Example 2, $X_c^1 = (x_1,x_2)$ is assigned to the first level. Therefore, $f(X) = U(1,x_4,x_3,0;x_1,x_2)$. If $X_c^2 = (x_3,x_4)$ is assigned to the first level, then $f(X) = U(U(1,0,0,0;\ X_c^1),\ U(1,1,0,0;\ X_c^1),\ U(1,0,1,0;\ X_c^1),\ U(1,1,1,0;\ X_c^1);\ X_c^2)$.

Based on the above theorems, we shall present the algorithm for synthesizing an arbitrary function of n variables.

1) The Case of One Control Variable: First consider the synthesis using T-ULM's of one control variable, i.e., $k = 1$, where $r \neq 2$. If the variables for which "dependency" becomes maximum is assigned to the first level, then it is obvious by Theorem

TABLE III
TRUTH TABLE OF EXAMPLE 2

x_1,x_2 \ x_3,x_4	00	01	10	11
00	1	1	1	1
01	0	0	1	1
10	0	1	0	1
11	0	0	0	0

5 that the total sum of "dependencies" for the other variables becomes minimum. By Theorem 6, the smaller l is, the larger value of "dependency" can be admitted in one U_l of the lth level. Therefore, it is considered that the number of T-ULM's necessary to realize residue functions of the successive levels may be smaller and that U_1 is fully used.

Once the variable for the first level is determined, the number r of the residue functions are decided. Similarly, at the next level, this procedure is continued. If the total sum of "dependency" s for the variables becomes zero, the successive T-ULM's are not necessary because the function is realized only with a true input or a constant function.

The above mentioned procedure is summarized in Subalgorithm 2.

Subalgorithm 2 (the Case of One Control Variable):

Step 1: For one of the variables, "dependency" of the given function $f(X)$ is calculated.

Step 2: After all the variables are checked, the variable x_i for which "dependency" becomes maximum are selected. If there exist some maximum ones, take one of them.

Step 3: This x_i is assigned to the first level of the network of T-ULM's. Each residue function is calculated as $f_j(X') = f(X', x_i \leftrightarrow j)$.

Step 4: The above procedure is applied to each residue function up to the bth level, where b is equal to the number of variables.

Step 5: By Lemma 1 (the case where $g(x) = x$) and Lemma 2, the reduction of the number of T-ULM's is carried out.

Example 4: Consider the synthesis of the ternary function of Table IV using T-ULM's of one control variable, "Dependencies" $N^f(x_1)$,$N^f(x_2)$, and $N^f(x_3)$ become 8, 2, and 5, respectively. Therefore, the total sum of "dependencies" is 15. Since $N^f(x_1)$ is maximum, x_1 is assigned to the first level. Similarly, in each residue function the same procedure is continued. As a result, $f(X)$ is expressed as

$$f(X) = U(U(0, 1, U(2, 0, 1; x_2); x_3), x_3,$$
$$\downarrow\ \downarrow \quad\ \downarrow$$
$$8\ 2 \quad\ 1$$
$$U(1, 2, U(2, 1, 0; x_2); x_3); x_1)$$
$$\downarrow \quad\ \downarrow$$
$$3 \quad\ 1$$

The total sum of "dependencies" 15 is decomposed as shown below the arrow.

2) *The Case of Multiple Control Variables:* Next consider the synthesis using T-ULM's of multiple control variables, i.e., $k \geq 2$. By Theorem 6, the smaller l is, the larger becomes the allowed value of "dependency" in one U_l. Therefore, it is desirable that to the higher levels the combinations of the variables with small dependencies are assigned.

In order to obtain such a desirable solution, the following technique is presented. First consider the case where the number of input variables is greater than or equal to $2k$. For each combination of k variables from n variables, "dependency" is calculated. This possible number of combinations is given as ${}_nC_k$. Arrange the number ${}_nC_k$ of control vectors according to the value of "dependency" so that the maximum one is in the last. From the variables for which "dependency" is minimum, pick up different $(n - k)$ variables in order. So, the other k variables are determined and they are assigned to the first level of a network.

TABLE IV
TRUTH TABLE OF EXAMPLE 4

x_1,x_2 \ x_3	0	1	2
00	0	1	2
01	0	1	0
02	0	1	1
10	0	1	2
11	0	1	2
12	0	1	2
20	1	2	2
21	1	2	1
22	1	2	0

Next consider the case where the number of input variables is less than $2k$. By Lemma 6, if X_c^i for which "dependency" becomes minimum is selected and remaining other variables are assigned to the first level, then the minimum realization is obtained.

Once the control vector of the first level which has a elements ($a \leq k$) is determined, then find the residue functions. As to each residue function, replace the number of input variables $n - a$, and the same procedure is continued.

Thus, the configuration of the control vectors can completely be determined, and a suboptimal solution can be obtained. This procedure is considered to be the extension of the case of $k = 1$, and is summarized in Subalgorithm 2.

Subalgorithm 2 (the Case of Multiple Control Variables)

Step 1: If the number of the input variables is less than $2k$, go to Step 8.

Step 2: Pick up k variables from the given n variables. The number of possible combinations is ${}_nC_k$.

Step 3: For one of these combinations, "dependency" of the given function $f(X)$ is calculated.

Step 4: After all the combinations are checked, arrange the control vectors according to the value of "dependency."

Step 5: For the variables for which "dependency" is smaller, pick up $n - k$ variables.

Step 6: The remaining other variables X_c^i are assigned to the first level of a T-ULM network.

Step 7: Each residue function is calculated as $f_j = f(X', X_c^i \leftrightarrow j)$. Go to Step 11.

Step 8: Pick up k or $n - k$ variables.

Step 9: For one of these combinations, "dependency" of the given function $f(X)$ is calculated.

Step 10: After all the combinations are checked, select the variables for which "dependency" becomes minimum. Go to Step 6.

Step 11: The above procedure is applied to each residue function up to the bth level.

Step 12: By Lemmas 1 and 2, the reduction of the number of T-ULM's is carried out.

Example 5: Consider the synthesis of the binary function of Table V using T-ULM's of two control variables. Each "dependency" becomes such as below.

$$N^f(x_1,x_2) = 9,\ N^f(x_1,x_3) = 9,\ N^f(x_1,x_4) = 6,\ N^f(x_1,x_5) = 10,$$
$$N^f(x_1,x_6) = 7,\ N^f(x_2,x_3) = 9,\ N^f(x_2,x_4) = 7,\ N^f(x_2,x_5) = 10,$$
$$N^f(x_2,x_6) = 7, N^f(x_3,x_4) = 10, N^f(x_3,x_5) = 10, N^f(x_3,x_6) = 9,$$
$$N^f(x_4,x_5) = 7,\ N^f(x_4,x_6) = 7,\ N^f(x_5,x_6) = 4.$$

The smallest one is $N^f(x_5,x_6)$. The next smallest one is $N^f(x_1,x_4)$.

TABLE V
TRUTH TABLE OF EXAMPLE 5

$x_1x_2x_3$	x_6 x_5 x_4							
		0	0	0	0	1	1	1	1
		0	0	1	1	0	0	1	1
		0	1	0	1	0	1	0	1
0 0 0		0	0	0	0	0	1	0	1
0 0 1		0	0	0	0	0	0	0	0
0 1 0		1	1	0	0	1	1	0	0
0 1 1		0	1	1	0	0	0	1	1
1 0 0		0	1	1	1	0	1	1	1
1 0 1		1	1	1	1	1	1	1	1
1 1 0		0	1	1	1	0	1	1	0
1 1 1		0	1	0	1	1	1	1	1

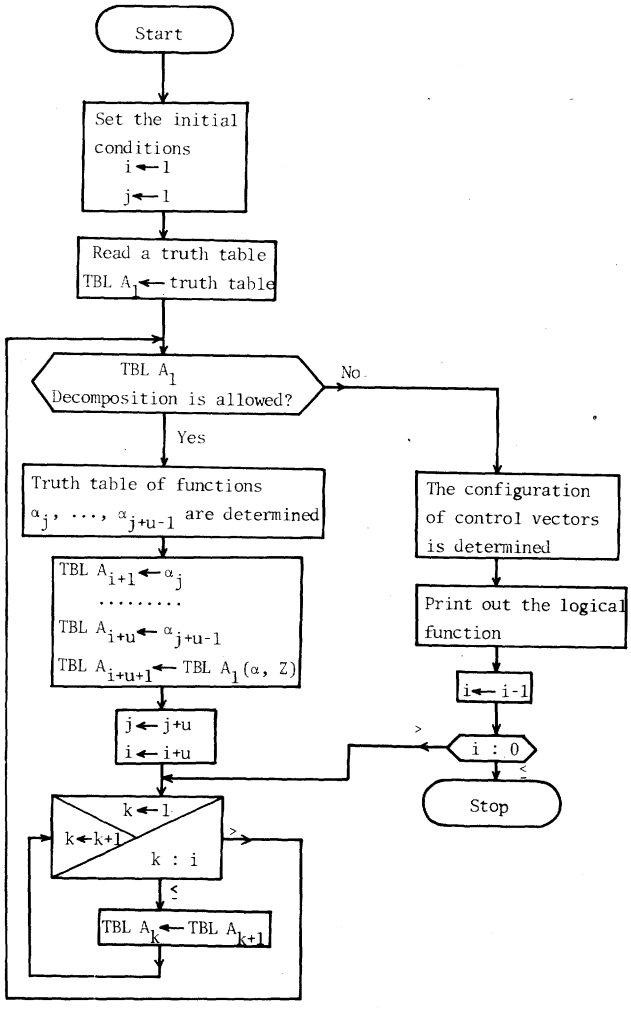

Fig. 4. Flow chart of synthesizing a T-ULM network.

So, the remaining (x_2,x_3) is selected as the control vector of the first level. Thus, the function $f(X)$ is realized such as below.

$$f(X) = U(U(0,x_5,x_6,1;x_1,x_4),\ U(1,x_4,0,U(1,1,1,0;x_4,x_6);\ x_1,x_5),\ x_1,U(U(0,1,1,0;\ x_4,x_5),\ x_4,x_5,1;\ x_1,x_6);x_2,x_3).$$

Now, we can systematically synthesize an arbitrary function of n variables by Subalgorithms as shown in the flow chart of Fig. 4.

IV. CONCLUSION

In this correspondence, we have presented the mathematical properties of T-ULM's in the multiple-valued logic. On the basis of these properties an algorithm for synthesizing a network of T-ULM's with a smaller number of T-ULM's has been discussed. The algorithm consists of the complex disjunctive decomposition and the proper order of the expansion. In order to evaluate such things, the decomposition is considered from the view point of the upper bound of the number of T-ULM's. Moreover, the proper order of the expansion is attributed to how to distribute "dependencies." As a result, it becomes clear that especially in the function of large variables, the algorithm becomes a useful method for the reduction of the number of T-ULM's.

However, the algorithm does not always give the optimal solution mainly because the method for using identical residue functions is not taken into consideration. Furthermore, we must consider the case where some unary functions of input variables besides true inputs are allowed.

ACKNOWLEDGMENT

The authors wish to thank Prof. T. Anayama for his guidance and encouragement.

REFERENCES

[1] S. S. Yau and C. K. Tang, "Universal logic circuits and their modular realizations," in *1968 Spring Joint Comput. Conf., AFIPS Proc.*, vol. 32. Washington, DC: Thompson, pp. 297–305, 1968.
[2] S. S. Yau and M. Orsic, "Synthesis of universal logic modules," in *Proc. 3rd Ann. Princeton Conf. on Inform. Sci. and Syst.*, pp. 498–503, 1969.
[3] S. S. Yau and C. K. Tang, "Universal logic modules and their applications," *IEEE Trans. Comput.*, vol. C-19, pp. 141–149, Feb. 1970.
[4] F. P. Preparata and D. E. Muller, "Generation of near-optimal universal boolean functions," *J. Comput. and Systems Sci.*, vol. 4, pp. 93–102, Apr. 1970.
[5] E. L. Post, "Introduction to a general theory of elementary propositions," *Amer. J. Math.*, vol. 43, pp. 163–185, 1921.
[6] R. D. Berlin, "Synthesis of N-valued switching circuits," *IRE Trans. Electron. Comput.*, vol. EC-7, pp. 52–56, Mar. 1958.
[7] D. I. Porat, "Three-valued digital system," *Proc. IEE*, vol. 116, pp. 947–954, June 1969.
[8] A. Rose, "Sur les éléments universal trivalent de décision," *Comptes Rendus*, vol. 269, pp. 1–3, 1969.
[9] M. Orsic, "Multiple-valued logic modules," in *Proc. 1972 Int. Symp. Multiple-Valued Logic*, pp. 95–99.
[10] J. Loader, "Universal decision elements in multiple-valued logic," *Zeitschr. F. math. logik. und grundlagen d. math.*, vol. 18, pp. 205–216, 1972.
[11] J. Loader, "Second order and higher order universal decision elements in m-valued logic," in *Proc. 1975 Int. Symp. on Multiple-Valued Logic*, pp. 53–57, 1975.
[12] A. Rose, "Conditioned disjunction as a primitive connective for the m-valued propositional calculus," *Mathematische Annalen*, vol. 123, pp. 76–78, 1951.
[13] C. Y. Lee and W. H. Chen, "Several-valued combinational switching circuit," *AIEE Trans.*, vol. 75, pt. I, pp. 278–283, July 1956.
[14] S. Thelliez, "Note on the synthesis of ternary combinational networks using T gate operators," *Electron. Lett.*, vol. 3, pp. 204–205, May 1967.
[15] T. Higuchi and M. Kameyama, "Ternary logic system based on T-gate," in *Proc. 1975 Int. Symp. on Multiple-Valued Logic*, pp. 290–304, 1975.
[16] T. Higuchi and M. Kameyama, "Synthesis of multiple-valued logic networks based on tree-type universal logic modules," in *Proc. 1975 Int. Symp. on Multiple-Valued Logic*, pp. 121–130, 1975.
[17] R. M. Karp, "Functional decomposition and switching circuit design," *J. Soc. Indust. Appl. Math.*, vol. 11, pp. 291–334, June 1963.
[18] J. C. Muzio and D. M. Miller, "Decomposition of ternary switching functions," in *Proc. 1973 Int. Symp. on Multiple-Valued Logic*, pp. 156–165, 1973.
[19] H. Mine and S. Fujita, "Decomposition of multi-valued switching function and its application to threshold networks," *IECE Japan Trans.*, vol. 56-D, pp. 681–688, Dec. 1973.

Seniority Logic: A Logic for a Committee Machine

MARTIN OSBORNE

***Abstract*—A logic for a committee of perceptrons, called seniority logic, and a local adjustment algorithm for training a seniority committee are described. Like a majority committee, a seniority committee has the capacity to solve any two-class problem in which the classes are disjoint. Unlike a majority committee, a seniority committee may have members added during training, and a seniority committee is free to attain a size needed to solve a problem. In computer simulations on patterns with binary components a seniority committee has a higher recognition rate and needs fewer members than a majority committee. Solutions learned during training have high recognition rates on the training set and are reasonably free of bias.**

Manuscript received March 2, 1976; revised December 2, 1976.

The author is with the Department of Mathematical Studies, Southern Illinois University, Edwardsville, IL 62026.

***Index Terms*—Binary variables, capacity, committee machine, pattern recognition, perceptron, two-class classifier.**

I. INTRODUCTION

Early successes in training a single perceptron [8] encouraged the building of networks of perceptrons. Networks provide greater classification power than a single perceptron, but methods for building and training networks are not fully understood. Good results have been obtained with a particularly simple class of networks called a committee. Committees can be trained by extensions of the algorithms used to train a single perceptron. Of the various types of committees, a majority committee is probably the best known. Ablow and Kaylor [1] have shown that a majority committee can be constructed to solve any two-class pattern recognition problem in which the classes are disjoint; however, the construction is not a useful training algorithm, because the resulting committee may have as many members as there are patterns to be classified. Kaylor [2] has conjectured that in a two-class pattern recognition problem with disjoint classes and binary pattern components, there is a majority committee among the smallest committees capable of solving the problem. Kaylor's conjecture is without practical significance because the size of the smallest committee is not known in advance, and the algorithms for training a majority committee require that the size be fixed throughout training [3], [7]. Worse, it has been found that training is most successful when a majority committee is larger than necessary [5]. Finally, there are no convergence theorems for committee training algorithms similar to the well-known perceptron convergence theorem.

This correspondence introduces a new type of committee called a seniority committee and a local adjustment algorithm for training it. Like a majority committee, a seniority committee can be constructed to solve any two-class problem in which the classes are disjoint but with perhaps as many members as there are patterns. The algorithm for training a majority committee is designed for problems in which the pattern components are binary. Unlike a majority committee, the size of a seniority committee is not fixed throughout training; instead starting with a single member, a seniority committee adds new members as needed. As a result of Kaylor's conjective, in computer simulations, seniority committees were compared to majority committees and were found to perform better than majority committees of the same size. A disadvantage of using a seniority committee is that the training algorithm is sensitive to parameters whose values are determined by trial and error. Not surprisingly, we have failed to find a convergence theorem for a seniority committee's training algorithm.

II. SENIORITY COMMITTEE

This section begins with an informal definition of a seniority committee. The members on a seniority committee are ranked from one to the total number of members, and each member is one of two types. When categorizing a pattern, members of the first type vote -1 (Class A) or abstain, the members of the second type vote $+1$ (Class B) or abstain. If all members abstain, the committee decides Class B, otherwise the vote of the highest ranking member prevails. This voting procedure suggests the name seniority committee. Fig. 1 illustrates the decision boundary of a five member seniority committee. The pattern space is R^2 and each member, being a perceptron, is defined by a line. Beside each line is a number, an arrow, and either $+1$ or -1. The number indicates a member's rank, the arrow points to the region on which a member votes, and the $+1$ or -1 is a member's vote. The region of the plane assigned by the committee to Class A is shaded.

A formal definition of a seniority committee follows. Let MMBR_i, $i = 1, \cdots, p$ be the members of a seniority committee.